Computación y programación en R: Tema 3

David V. Conesa Guillén

GEeitEma

Valencia Bayesian Research Group

Dept. d'Estadística i Investigació Operativa

Universitat de València

Tema 3: Descripción numérica y gráfica de datos.

En este tema:

1.- Estadística básica.

2.- Distribuciones de Probabilidad. Generación de variables aleatorias.

3.- Tablas de frecuencias.

4.- Medidas de localización, dispersión y forma.

5.- Descripción gráfica de datos en R.

6.- Gráficos para datos discretos.

Tema 3: Descripción numérica y gráfica de datos.

También:

7.- Gráficos para datos continuos.

8.- Representación de datos multivariantes.

9.- Gráficos para estudiar la distribución de unos datos. Estimación de densidades.

10.- Parámetros gráficos. Cambios permanentes: la función `par()`.

11.- Exportando gráficos. Dispositivos gráficos (device drivers).

1.- Estadística básica.

- Los datos obtenidos cuando realizamos cualquier experimento presentan variabilidad:
  - el peso de un bebe al nacer varía
  - la cantidad de lluvia recogida en un dia en una determinada zona varía
  - la altura de una planta sometida a dos tipos de abono varía, etc.
- La Estadística es una disciplina que se ha desarrollado en respuesta a los experimentadores cuyos datos exhiben variabilidad.
- Los conceptos y métodos de la estadística nos permiten describir la variabilidad, planificar la investigación teniéndola en cuenta y analizar los datos para extraer el máximo de información de los mismos así como determinar la fiabilidad de las conclusiones que podamos obtener a partir de estos datos.
- Ya sabemos que `R` es un lenguaje que permite implementar técnicas estadísticas.
- En este tema vamos a ver cómo podemos utilizar `R` para analizar los bancos de datos que habitualmente se nos presentan al trabajar.

1.- Estadística básica.

- Variable ⟶ Característica de interés.
- Muestra Observada ⟶ Conjunto de valores de la variable observados obtenidos de manera homogénea.
- Tamaño muestral ⟶ Número de datos observados.
- La manera de describir la muestra (nuestros datos) depende del tipo de atributo:
  - Cualitativo ⟶ Intrínsecamente no tiene carácter numérico (categórica)
    - ⋆ Nominal (sin orden entre los valores): Sexo
    - ⋆ Ordinal (con valores ordenados): Nivel de estudios
  - Cuantitativo ⟶ Intrínsecamente numérico
    - ⋆ Discreto (cantidad finita o numerable de valores): Número de hijos
    - ⋆ Continuo (valores en toda la recta real): Altura

2.- Distribuciones de Probabilidad.

- R tiene las distribuciones de probabilidad más comunes implementadas en la librería BASE. En otras librerías disponemos de otras tantas.
- Para cada una de ellas (`distrib`), disponemos de 4 versiones:

generador de numeros aleatorios	rdistrib
función densidad/probabilidad	ddistrib
función distribución	pdistrib
función inversa distribución (cuantiles)	qdistrib

Ejemplo

```
x.norm <- rnorm(500) # Simulación de 500 datos normales
2*pt(-2.43, df = 13) # P-valor de dos colas de una t(13)
qf(0.99, 2, 7) # Percentil 99 de una distribución F(2, 7)
```

Distribuciones de probabilidad en la librería BASE.

Función	Utilidad
Normal	rnorm(n, mean=0, sd=1)
exponencial	rexp(n, rate=1)
gamma	rgamma(n, shape, scale=1)
Poisson	rpois(n, lambda)
Weibull	rweibull(n, shape, scale=1)
Cauchy	rcauchy(n, location=0, scale=1)
beta	rbeta(n, shape1, shape2)
t de Student	rt(n, df)
F (Snedecor)	rf(n, df1, df2)
Pearson χ^2	rchisq(n, df)
binomial	rbinom(n, size, prob)
geométrica	rgeom(n, prob)
hypergeométrica	rhyper(nn, m, n, k)
logística	rlogis(n, location=0, scale=1)
lognormal	rlnorm(n, meanlog=0, sdlog=1)
binomial negativa	rnbinom(n, size, prob)
uniforme	runif(n, min=0, max=1)

3.- Tablas de frecuencias para variables categóricas.

- Hemos visto un factor es un vector utilizado para especificar una clasificación discreta de los elementos de otro vector de igual longitud,
- y que en R existen dos tipos de factores (variables categóricas):
  - Nominales: No existe jerarquía entre ellos (p.e., colores)
  - Ordinales: Existe jerarquía entre ellos (p.e., grupos de edad)
- Del mismo modo, dos factores definen una tabla de doble entrada, y así sucesivamente.
- La función table() calcula tablas de frecuencias a partir de factores de igual longitud.
- Si existen k argumentos categóricos, el resultado sería una variable k-indexada, que contiene la tabla de frecuencias.

Ejemplo

```
x <- as.factor(1:5)
table(x)
```

4.- Medidas de localización, dispersión y forma para variables cuantitativas continuas.

- La forma más sencilla de empezar a describir unos datos cuantitativos es realizar un resumen estadístico.
- En la página siguiente tenemos un listado de las más habituales para describir la localización y la dispersión.
- Para analizar la curtosis y la asimetría de unos datos podemos utilizar dos funciones de la librería `e1071`: `skewness()` y `kurtosis()`.

Ejemplo

```
x <- rgamma(50,1,3)
summary(x); fivenum(x)
mean(x); median(x); quantile(x); quantile(x,c(0.35,0.9))
sd(x); var(x); range(x); IQR(x)
min(x); which.min(x); x[which.min(x)]; pmin(x[1:5],x[6:10])
max(x); which.max(x); x[which.max(x)]; pmax(x[4:8],x[2:6])
```


Medidas de localización y dispersión más habituales.

Función	Utilidad
`sum(..., na.rm=FALSE)`	Suma
`max(..., na.rm=FALSE)`	Máximo
`min(..., na.rm=FALSE)`	Mínimo
`which.min(x)`	Posición del máximo
`which.max(x)`	Posición del mínimo
`pmax(...,na.rm=FALSE)`	Máximo en paralelo
`pmin(...,na.rm=FALSE)`	Mínimo en paralelo
`cumsum(x), cumprod(x)`	Sumas y prods acumulados
`cummax(x), cummin(x)`	max's y min's acumulados
`mean(x, trim=0, na.rm=FALSE)`	Media
`weighted.mean(x,w,na.rm=FALSE)`	Media ponderada
`median(x,na.rm=FALSE)`	Mediana
`quantile(x,prob=(0,0.25,0.5,0.75,1),na.rm=F)`	Cuantiles
`fivenum(x, na.rm=FALSE)`	5-Tukey: min, lower-hinge mediana, upper-hinge, máximo
`summary(x, na.rm=FALSE)`	min,1c,mediana,media,3c,max
`IQR(x, na.rm=FALSE)`	Rango inter-cuartílico
`range(...,na.rm=FALSE, finite=FALSE)`	Rango
`var(x, y=x, na.rm=FALSE, use)`	Varianza
`sd(x, na.rm=FALSE)`	Desviación Típica
`mad(x,center,constant=1.4426, na.rm=FALSE)`	Desviación mediana absoluta

Ejemplo

```
library(e1071)
x <- rgamma(50,1,3)
moment(x,2,center=F) # momento no centrado de orden 2
```

Consideramos dos distribuciones asimétricas (Betas) y las vamos a comparar con la normal que es simétrica:

```
nsim<-5000
s1<-skewness(rbeta(nsim,2,3))
s2<-skewness(rbeta(nsim,3,2))
s3<-skewness(rnorm(nsim,0.5,0.5))
s1;s2;s3
```

Consideramos ahora una distribución normal y una Student, más achatada, y las comparamos:

```
k1<-kurtosis(rnorm(nsim))
k2<-kurtosis(rt(nsim,3))
k1;k2
```


5.- Descripción gráfica de datos en R.

- Las gráficas son la mejor forma de simplificar lo complejo. Un buen gráfico suele ser más accesible que una tabla. Sin embargo es muy importante tener claro qué gráfico queremos hacer.
- Las facilidades gráficas de R constituyen una de las componentes más importantes de este lenguaje.
- R incluye muchas y muy variadas funciones para hacer gráficas estadísticas estándar: desde gráficos muy simples a figuras de gran calidad para incluir en artículos y libros.
- Permite además construir otras nuevas a la medida del usuario (aunque a veces hacer cosas *simples* no es fácil).
- Permite exportar gráficas en distintos formatos: PDF, JPEG, GIF, etc.
- Para ver una demo de gráficos con colores: `demo(graphics)`.
- Aquí únicamente veremos algunas de todas las posibilidades. Más detalles en el libro R Graphics de Paul Murrell.
- Otra alternativa: `ggplot`.

Gráficas en R

- R tiene dos sistemas de producir gráficos:
  - El *tradicional*, que es el que veremos principalmente
  - Gráficos *Trellis* (paquete `Lattice`) del que veremos algunos ejemplos
- Podemos dividir los comandos para efectuar las gráficas en tres grupos:
  - Funciones para crear gráficas de alto nivel, es decir ya programadas y que admiten diferentes posibilidades.
  - Funciones de bajo nivel, que permiten un control más fino del dibujo y permiten crear gráficas a medida.
  - Funciones para el uso interactivo, para extraer información de una gráfica o una modificación mediante el ratón.

Función `plot()`

El procedimiento gráfico de alto nivel más habitual para dibujar datos es `plot()`.

Ejemplo

```
x<-(0:65)/10
y<-sin(x)
plot(x)
plot(x, y)
plot(x, y, main="Función Seno")
z<-cos(x)
windows() #Crea una ventana nueva
plot(x, z, main="Función Coseno")
```

Opciones de la función plot()

Algunas de las más útiles

- `main`: Cambia el título del gráfico
- `sub`: Cambia el subtítulo del gráfico
- `type`: Tipo de gráfico (puntos, líneas, etc.)
- `xlab, ylab`: Cambia las etiquetas de los ejes
- `xlim, ylim`: Cambia el rango de valores de los ejes
- `lty`: Cambia el tipo de línea; `lwd`: Cambia el grosor de línea
- `col`: Color con el que dibuja

Ejemplo

```
plot(x, y, main="Seno", type="l")
plot(x, z, main="Coseno", lty=2, col="red", type="l")
plot(x, z, main="Coseno", lty=3, col="blue", type="l",
xlim=c(0, 2), ylab="cos(x)")
```


Procedimientos de bajo nivel

Hay una serie de funciones que permiten dibujar sobre una gráfica ya creada.

Los más habituales

- `points(x, y, ...)`: Dibuja una nube de puntos
- `lines(x, y, ...)`: Dibuja una línea que une todos los puntos
- `ablines()`: Dibuja una línea recta dada la interc. y pendiente
- `polygons(x, y, ...)`: Dibuja un polígono cerrado
- `text(x, y, labels, ...)`: Escribe texto en unas coordenadas

Ejemplo

```
plot(x, y, main="Funciones seno y coseno", type="l")
lines(x, z, col="blue", lty=2) # col=4 es equivalente
text(x=c(0.5, 0.5), y=c(0, 1), labels=c("sin(x)", "cos(x)"),
col=c("black", "blue"))
```

Leyendas

Descripción

La función `legend(x, y, legend, ...)` permite añadir leyendas a un gráfico:

- `x,y` : Esquina sup. izda. de la leyenda
- `legend`: Texto de la leyenda
- `bty`: Tipo de borde ("n" para omitir)

Ejemplo

```
plot(x, y, main="Funciones seno y coseno", type="l")
lines(x, z, col="blue", lty=2)
legend(x=3, y=1, legend=c("sin(x)", "cos(x)"), lty=c(1,2),
col=c("black", "blue"))
```


Funciones gráficas interactivas.

En R existen una serie de funciones que permiten completar los gráficos de manera interactiva por parte del usuario

Descripción

- `identify(x, y, etiquetas)` identifica los puntos con el ratón y escribe la correspondiente etiqueta.
- `locator()` devuelve las coordenadas de los puntos.

Ejemplo

```
plot(x, y, main="Funciones seno y coseno", type="l")
lines(x, z, col=2, lty=2)
legend(locator(1),legend=c("sin(x)","cos(x)"),lty=c(1,2),col=c(1,2))

x <- 1:10; y <- sample(1:10)
nombres <- paste("punto", x, ".", y, sep ="")
plot(x, y); identify(x, y, labels = nombres)
```

6.- Gráficos para datos discretos.

Los más habituales

Para representar variables categóricas o cuantitativas discretas (con pocas clases):

- Diagramas de puntos: `dotplot()`
- Diagramas de barras: `barplot()`
- Diagramas de quesos: `pie()`

Ejemplo

```
library(lattice)
x <- rbinom(100,5,0.3)
dotplot(table(x),horizontal=F)
par(mfrow=c(2,2)); plot(x,type="h") # Diagrama de puntos
barplot(table(x),col=rainbow(length(table(x)))) # Diagrama barras
pie(table(x)) # Diagrama de quesos
```


7.- Gráficos para datos continuos.

Los más habituales

- Diagramas de cajas: `boxplot()`
- Diagramas de tallo y hojas: `stem()`
- Diagramas de puntos: `stripchart()`

Ejemplo

```
y <- rnorm(100); y.f <- rbinom(100,5,0.3)
# Gráfico de tallos y hojas
stem(y)
par(mfrow=c(2,2)); m<-mean(par("usr")[1:2]) # medidas ventana usuario
# Diag. de cajas
boxplot(y); boxplot(y~y.f); boxplot(split(y,y.f),col="cyan")
# diagrama de puntos, tres métodos
stripchart(y); text(m, 1.04, "stripchart método overplot")
stripchart(y,method="jitter",add=T,at=1.2); text(m,1.35,"método jitter")
stripchart(round(y,1),method="stack",add=T,at=0.7); text(m,0.85,"método
stack")
```

8.- Representación de datos multivariantes.

Cuando queremos representar varias variables conjuntamente para detectar relaciones entre ellas, disponemos de diversos tipos de gráficos:

Los más habituales

- Gráficos de tendencias para tablas de contingencia: `dotchart()`
- Gráficos de dispersión: `plot()` y `pairs()`
- Gráficos condicionados: `coplot()`.

Ejemplo

```
# Gráficos de tendencias para tablas de contingencia
data(VADeaths)
dotchart(VADeaths, main = "Death Rates in Virginia - 1940")
# Gráficos condicionados
data(quakes)
coplot(lat~long | depth, data = quakes)
```


Ejemplo

```
# Gráficos de dispersión para revisar relaciones entre variables
X <- matrix(rnorm(1000), ncol = 2); colnames(X) <- c("a", "b")
plot(X)

X <- matrix(rnorm(1000), ncol = 5)
colnames(X) <- c("a", "id", "edad", "loc", "peso")
pairs(X)

data(iris)
razas<-unclass(iris$Species)
plot(iris[1:2],pch=21,bg = c("red", "green3", "blue")[razas])
pairs(iris[1:4], main = "Anderson's Iris Data - 3 species", pch = 21, bg
= c("red", "green3", "blue")[razas])

data(swiss)
pairs(swiss, panel = panel.smooth, lwd = 2, cex= 1.5, col="blue")
```

9.- Gráficos para estudiar la distribución de unos datos.

Cuando queremos estudiar cual es la posible distribución de unos datos disponemos de diferentes funciones:

Los más habituales

- Histogramas: `hist()`
- Gráficos *qq*: `qqplot()`, `qqnorm()` y `qqline()`. Dos posibles usos:
  - Comparación de cuantiles empíricos versus cuantiles teóricos: para comprobar si los datos se parecen a una determinada distribución
  - Comparación de dos distribuciones empíricas entre sí
- Estimación de la función de distribución empírica: `ecdf()`
- Estimación kernel de la función de densidad: `density()`

Ejemplo

```
# Histogramas
y<-rnorm(500); hist(y); hist(y,5)
```


Gráficos qq

Ejemplo

```
y<-rnorm(500)
# Comparación de los cuantiles muestrales con los de una Normal
qqnorm(y); qqline(y)
# Comparación de los cuantiles muestrales de dos muestras
y.t<-rt(500,3)
qqplot(y,y.t,xlab="Dist. Normal", ylab="Dist. St(3)"); qqline(y)
# Comparar cuantiles muestrales con los de una distribución dada
library(lattice)
qqmath(y,distribution=function(p){qt(p,df=5)})
qqmath(y,distribution=function(p){qgamma(p,shape=3,rate=5)})
```

Función de distribución empírica.

Ejemplo

```
x<-rlogis(500,2,3)
Fn.x<-ecdf(x); summary(Fn.x)
plot(Fn.x,main="Función Distribución Empírica")
plot(Fn.x,add=T,verticals=T,col.v=2,col.h=4)
x.seq<-seq(min(x),max(x),length=4)
# Para evaluar Fn en cualquier punto
points(x.seq,Fn.x(x.seq),pch=21,bg="red3")
abline(v=x.seq,col="red3")
abline(h=Fn.x(x.seq),col="red3")
```


Estimación de funciones de densidad.

Ejemplo

```
x<-rgamma(500,4,3)
hist(x,prob=T) # prob=T equivale a freq=F

# Estimador kernel de la densidad
lines(density(x))
bw.x<-density(x)$bw
bw.x # amplitud de la banda
# Podemos modificar la banda para la estimación
lines(density(x,bw=bw.x/2),col=2)
lines(density(x,bw=bw.x*2),col=4)
```

Representación en 3D.

Cuando queremos representar una función bivariante disponemos de diversos tipos de gráficos:

Los más habituales

- Gráficos en tres dimensiones: `image()`
- Gráficos de contorno: `contour()`. Permite añadir líneas de nivel.
- Las librerías `MASS` y `ks` tienen funciones para estimar kernels bivariantes.

Ejemplo

```
x<-seq(-1,1,0.05); y<-seq(-1,1,0.05)
f <- function(x, y) cos(y)/(1 + x^2)
z <- outer(x, y, f)
image(x,y,z); contour(x,y,z,add=T)
```


10.- Parámetros gráficos. La función `par()`.

- Al igual que en la función `plot()`, podemos controlar casi todos los aspectos de una gráfica mediante los parámetros gráficos.
- `R` dispone de una lista (unos 70) de parámetros gráficos que controlan, por ejemplo, el color, tipo de línea, grosor, tipo de punto, justificación del texto, tamaño, etc.
- Vale la pena leer con calma la ayuda de `par`.
- Cada parámetro tiene un nombre (p.e. "col" para color) y un valor. `R` dispone de colores de los más variado. Ver el archivo `UsingColorInR`.
- Cada vez que hacemos una gráfica se abre un dispositivo gráfico con una lista de parámetros que tiene unos valores iniciales por defecto.
- Los parámetros gráficos pueden fijarse:
  - de forma permanente para ese dispositivo mediante la función `par()`
  - o temporalmente en las llamadas a las funciones gráficas incluyéndolos en la lista de argumentos (si lo permiten).

Funcionamiento de la función `par()`*.*

- Sin argumentos devuelve una lista con los parámetros y sus valores en activo.

Ejemplo

```
par()
```

- Con un argumento, vector de caracteres, con los nombres de algunos parámetros, devuelve una lista con los parámetros y sus valores en activo.

Ejemplo

```
par(c("col","lty"))
```

- Con nombres de parámetros = valor, establece los nuevos valores

Ejemplo

```
par(col=4,lty=2)
```


Ejemplos de parámetros gráficos

Colocar varias gráficas en una ventana

Los siguientes parámetros permiten diseñar el número de gráficas en cada dispositivo gráfico

- `mfrow`: Nº de filas y columnas en la ventana. Los huecos se rellenan por filas.
- `mfcol`: Ídem pero se rellena por columnas.

Ejemplo

```
x<-(0:65)/10
y<-sin(x)
par(mfrow=c(1,2)) # probar también c(2,1)
plot(x, y, main="Seno", type="l", ylab="sin(x)")
plot(x, z, main="Coseno", type="l", lty=2, col="red",
ylab="cos(x)")
```

11.- Dispositivos gráficos (device drivers).

- El funcionamiento de los diferentes dispositivos gráficos depende mucho del entorno de trabajo.
- Por defecto, cuando realizamos la primera gráfica, `R` abre un dispositivo gráfico.
- Podemos abrir nuevas ventanas gráficas (p.e. en el entorno Windows llamando a la función `windows()`). Con ello tendremos varios dispositivos donde dibujar.
- Para cerrar un dispositivo abierto utilizamos `dev.off()`. Si no tenemos claro cual cerrar, la función `dev.list()` nos puede ayudar a saber qué dispositivos hay abiertos y que numeración tienen.
- Siempre hay uno activo, podemos saber cuál es con `dev.cur()`. Si queremos activar otro podemos utilizar `dev.set()`.
- Con la opción histórico grabando activa `R` nos permite disponer de todos los gráficos e ir accediendo al resto con Av.Pág. y Re.Pág.
- RStudio permite una gestión muy sencilla de los gráficos realizados.

Exportando gráficos.

- Para guardar una gráfica, podemos copiar y pegar desde la ventana gráfica a un tratamiento de textos que los permita.
- Desde el menú `Archivo -> Guardar como` podemos guardar la gráfica como un fichero *metafile*, *pdf*, *png*, *bmp*, *postscript*, *tif* o *jpg*.
- Sin embargo esta opción no es la mejor ya que no tenemos control sobre la propia gráfica y como queda guardada. Sobre todo a nivel de escala.
- Lo mejor es enviar directamente la gráfica a un dispositivo (pdf, postscript, etc.) utilizando funciones como `pdf()` o `postscript()`.

Ejemplo

```
pdf("prueba.pdf", paper="special", width=13, height=7)
hist(x<-rnorm(100),prob=T)
dev.off()
```

Licencia de este material